CLAIMS

What is claimed is:

1. A computer system, comprising:

a pipelined, simultaneous and redundantly threaded ("SRT") processor comprising at least one data cache;

an I/O controller coupled to said processor, which in turn is coupled to at least one I/O device;

a main system memory coupled to said processor; and

wherein said SRT processor processes a set of instructions in a leading thread and also in a redundant trailing thread to detect transient faults in the computer system; and

wherein when a data load command appears in the leading thread, the processor loads the requested data and replicates the value for the corresponding data load command in the trailing thread.

2. The computer system of claim 1 further comprising a load value queue;

wherein when the processor loads the requested data, the processor stores the same data in the load value queue.

3. The computer system of claim 2 wherein the processor does not store the data value in the load value queue until the data load command in the leading thread commits.

4. The computer system of claim 2 wherein the load value queue is a FIFO buffer and wherein all data load commands in the trailing thread are executed by the processor in their original, program order.

5. The computer system of claim 2 wherein the data load command is a request for data in the data cache.

6. The computer system of claim 2 wherein the data load command is a request for data in the main system memory.

7. The computer system of claim 4 wherein if the load value queue becomes full, execution of instructions in the leading thread is temporarily halted to prevent more data values from entering the load value queue; and

wherein if the load value queue becomes empty, execution of instructions in the second thread is temporary halted to allow more data values to enter the load value queue.

8. A pipelined, simultaneous and redundantly threaded ("SRT") processor, comprising:

a program counter configured to assign program count identifiers to instructions in each thread that are retrieved by the processor;

floating point execution units configured to execute floating point instructions;

integer execution units configured to execute integer-based instructions;

load/store units configured to perform fetch and store operations to or from data sources such as a data cache, memory, and data registers; and wherein said processor is configured to detect transient faults during program execution by executing instructions in at least two redundant copies of a program thread and wherein false errors caused by incorrectly replicating fetched data in the redundant program threads are avoided by replicating the actual data retrieved from data fetch instructions in a first program thread for a second program thread.

9. The SRT processor of claim 8 wherein the processor further comprises:

a load value queue for storing the data values fetched in response to data fetch instructions in the first program thread;

wherein the load/store units place a duplicate copy of the data in the load value queue after fetching the data from the data source and wherein the load/store units access the load value queue and not the data source to fetch data values in response to data fetch instructions in the second program thread.

10. The SRT processor of claim 9 further comprising a register update unit;

wherein the register update unit is configured to hold instructions in a queue until the instructions are executed and retired by the SRT processor and wherein the data fetched by the load/store units in response to data fetch instructions in the first program thread is not placed in the load value queue until the data fetch instructions in the first program thread retire from the register update unit.

11. The SRT processor of claim 9 wherein the SRT processor is an out-of-order processor capable of executing instructions in the most efficient order, but wherein data fetch instructions are executed in the same order in both the first and second program threads.

12. The SRT processor of claim 11 wherein the load value queue is a FIFO buffer and data is transmitted to and from the buffer using an error correction technique.

13. The SRT processor of claim 12 wherein the individual load value entries in the load value queue comprise:

the size of the fetched data value;

an address indicating the physical location in the data source from which the data was fetched; and

the data value that was retrieved by the load/store units when the fetch command was executed.

14. The SRT processor of claim 12 wherein if the load value queue becomes full, the first thread is stalled to prevent more data values from entering the load value queue; and

wherein if the load value queue becomes empty, the second thread is stalled to allow data values to enter the load value queue.

15. A method of replicating data values in an SRT processor which can fetch and execute a program set in two separate threads so that each thread includes substantially the same instructions

as the other thread, one of said threads being a leading thread and the other of said threads being a trailing thread, the method comprising:

accessing a data source to load a data value when the leading thread requests said data;

storing the data value in a load value queue;

accessing the load value queue for the data value for corresponding data requests in the trailing thread.

16. The method of claim 15 further comprising:

executing the data requests in the trailing thread in program order.

17. The method of claim 16 further comprising:

storing the data values in the load value queue after the data requests in the leading thread execute.

18. The method of claim 17 further comprising:

storing the data value in a FIFO buffer; and

storing the following information with each entry in the buffer:

the size of the data value;

an address indicating the physical location in the data source from which the data was fetched; and

the data value that was retrieved by the data request in the leading thread.

19. The method of claim 18 wherein the data source is a data cache.

20. The method of claim 18 wherein the data source is system memory source.

21. The method of claim 17 further comprising:

transmitting data to and from the load value queue using an error correction technique.